BINARY SEARCH TREE INSERTION, THE HYPOPLACTIC INSERTION, AND DUAL GRADED GRAPHS

JANVIER NZEUTCHAP

Abstract. Fomin (1994) introduced a notion of duality between two graded graphs on the same set of vertices. He also introduced a generalization to dual graded graphs of the classical Robinson-Schensted-Knuth algorithm. We show how Fomin's approach applies to the binary search tree insertion algorithm also known as sylvester insertion, and to the hypoplactic insertion algorithm.

Contents

1. Introduction and definitions

The Young lattice is defined on the set of partitions of positive integers, with covering relations given by the natural inclusion order. The differential poset nature of this graph was generalized by Fomin with the introduction of graph duality [12]. With this extension he introduced [14] a generalization of the classical Robinson-Schensted-Knuth [2, 3] algorithm, giving a general scheme for establishing bijective correspondences between pairs of saturated chains in dual graded graphs, both starting at a vertex of rank 0 and having a common end point of rank n, on the one hand, and permutations of the symmetric group $\mathfrak{S}_n$ on the other hand. This approach naturally leads to the Robinson-Schensted insertion algorithm.

Roby [16] gave an insertion algorithm, analogous to the Schencted correspondence, for mapping a permutation to a pair of Young-Fibonacci tableaux, interpreted as saturated chains in the Fibonacci lattice $Z(1)$ introduced by Stanley [11] and also by Fomin [13], and he showed that Fomin's approach is partially equivalent to his construction. He also [17] made a connection between graded graphs and the Robinson-Schensted correspondence for skew oscillating tableaux. More recently, Cameron and Killpatrick [10] gave an insertion algorithm, analogous to the Schensted correspondence, for mapping a colored permutation to a pair of domino Fibonacci tableaux, interpreted as saturated chains in the Fibonacci lattice $Z(2)$, and they also showed that Fomin's approach is partially equivalent to their construction. For both constructions in $Z(1)$ and $Z(2)$, an evacuation is needed to make the insertion algorithm coincide with Fomin's approach [9, 10].

The motivation of this note is to do the same for two other combinatorial insertion algorithms, namely the binary search tree insertion algorithm of Knuth [4] also known as the sylvester insertion

2000 *Mathematics Subject Classification.* Primary 05-06; Secondary 05E99.

Key words and phrases. Graded graphs, Robinson-Schensted, Fomin's approach, hypoplactic, sylvester, binary search tree.

as defined by Hivert et al [6], and the hypoplactic insertion algorithm of Krob and Thibon [5]. In section 1.1 we recall the necessary background and definitions on graph duality ; the reader should refer to [12] for more details on the subject. In section 2 we first recall the hypoplactic insertion algorithm, then we build isomorphic images of two dual graded graphs introduced by Fomin, and define a growth function or 1-correspondence into those graphs. We end the section with the application of Fomin's approach using that correspondence into the two graphs, and we relate the pairs of tableaux obtained from the hypoplactic insertion algorithm to the pairs of tableaux obtained from Fomin's growth diagrams. In section 3 we perform a similar process for the binary search tree insertion (BST). The results presented in this note were first announced in [7].

1.1. **Definitions.**

Definition 1.1 (graded graph)**.** *A graded graph is a triple $G = (P, \rho, E)$ where P is a discrete set of vertices, $\rho : P \to \mathbb{Z}$ is a rank function and E is a multi-set of edges (x, y) satisfying $\rho(y) = \rho(x)+1$.*

Let $G = (G_1, G_2) = (P, \rho, E_1, E_2)$ be a pair of graded graphs with a common set of vertices and a common rank function, where E_1 is the set of G_1-edges, directed upwards (in the direction of increasing rank) and E_2 the set of G_2-edges, directed downwards (in the direction of decreasing rank).

Let $\mathbb{K}$ be a field of characteristic zero, define $\mathbb{K}P$ as the vector space formed by linear combinations of vertices of P. One can now define two linear operators U (Up) and D (Down) acting on $\mathbb{K}P$ as follows.

$$U\,x \;=\; \sum_{(x,y)\,\in\,E_1} w_1(x,y)\; y \qquad ; \qquad D\,y \;=\; \sum_{(x,y)\,\in\,E_2} w_2(x,y)\; x \tag{1.1}$$

where $w_i(x, y)$ is the multiplicity or the weight of the edge (x, y) in E_i.

Definition 1.2 (graph duality)**.** *G_1 and G_2 are said to be dual* [12] *if the two operators U and D satisfy the commutation relation below.*

$$D_{n+1}\,U_n \;=\; U_{n-1}\,D_n + I_n \tag{1.2}$$

where U_n (resp. D_n) denotes the restriction of the operator U (resp. D) to the n^{th} level of the graph, and I_n the identical operator at the same level. There are generalizations of this definition, notably the case of an r-duality with $r > 1$ and the case of an r_n-duality, with the relations bellow.

$$D_{n+1}\,U_n \;=\; U_{n-1}\,D_n + r\,I_n \quad \textit{and} \quad D_{n+1}\,U_n \;=\; U_{n-1}\,D_n + r_n\,I_n \tag{1.3}$$

A well-known example is the Young lattice of partitions of integers, which is a *self-dual* graded graph ($G_1 = G_2$) or *differential poset* [11]. Its self-duality expresses the fact that for any Ferrers diagram λ, there is one more Ferrers diagram obtained by adding a single box to λ than by deleting a single box from λ, and for any couple of Ferrers diagram (λ, μ) there are as many Ferrers diagram simultaneously contained by λ and μ than those simultaneously containing λ and μ.

Definition 1.3 (1-correspondence)**.** *Introduced by Fomin* [12]*, it denotes any bijective map ϕ in a self-dual graded graph $G = (P, \rho, E)$, sending any pair (b_1, b_2) of edges having a common end point, to a triple (a_1, a_2, α) where a_1 and b_1 are two edges of the lattice having a common start point, α is either 0 or 1, and the following properties are satisfied, where a_1 is the edge (t, x), $a_2 \overset{def}{=} (t, y)$, $b_1 \overset{def}{=} (y, z)$ and $b_2 \overset{def}{=} (x, z)$.*

1. *$end(a_1) = start(b_2)$ and $end(a_2) = start(b_1)$;*
2. *if b_1 and b_2 are degenerated, that is to say there exists a vertex $x_0 \in P$ such that $b_1 = b_2 = (x_0, x_0)$, then $(a_1, a_2, \alpha) = (b_1, b_2, 0)$.*

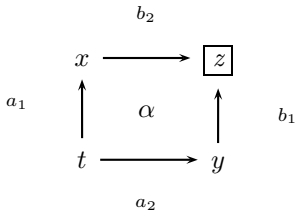


FIGURE 1. A square in a growth diagram.

A 1-correspondence can be used to build a *growth diagram*, which is a sort of pictorial representation of a family of correspondences similar to the Schensted correspondence. This approach is introduced by Fomin [14] and concerns a family of insertion algorithms sending permutations onto pairs of labeled combinatorial objects *of the same shape*, interpreted as saturated chains in the differential poset considered.

For any permutation σ, the growth diagram $d(\sigma)$ is build the following way. First draw the permutation matrix of σ ; next fill the left and lower boundary of $d(\sigma)$ with the empty combinatorial object generally denoted by $\emptyset$. The rest of the construction is iterative ; $d(\sigma)$ is filled from its lower left corner to its upper right corner, following the diagonal. At each step and for any configuration (a_1, a_2, α) as pictured above (Fig. 1), z is obtained by application of the 1-correspondence to t, x, y and α.

2. Fomin's approach applied to the hypoplactic insertion algorithm

Introduced in [5], the hypoplactic correspondence is an insertion algorithm, analogous to the Robinson-Schensted correspondence, mapping a permutation to a pair made of a quasi-ribbon tableau and a ribbon tableau. It appears in the study of noncommutative symmetric functions. A ribbon tableau is a composition diagram filled with positive integers in such a way that entries increase across lines from left to right, and up columns. In a quasi-ribbon diagram, entries increase down columns. As usual, the insertion tableau of a permutation is iteratively constructed reading its letters from left to right. To insert a letter **a** in a quasi-ribbon, compare **a** with the last letter z in its last row. If **a** is greater then just append it to the right of z. Otherwise, reading the quasi-ribbon from left to right and top to bottom, find the last entry y such that $y \leq \mathbf{a}$, then insert a cell labeled **a** just to the right of the one labeled y and shift the rest of the quasi-ribbon below the newly created cell. Let us apply the algorithm to the permutation $\sigma = 415362$.

Example 2.1.

$$\boxed{4^1} \rightarrow \begin{array}{|c|}\hline 1^1 \\ \hline \mathbf{4^2} \\ \hline \end{array} \rightarrow \begin{array}{|c|c|}\hline 1 & \\ \hline 4 & 5^1 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline 1 & 3^1 & \\ \hline & \mathbf{4^2} & \mathbf{5^2} \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|}\hline 1 & 3 & & \\ \hline & 4 & 5 & 6^1 \\ \hline \end{array} \rightarrow P(\sigma) = \begin{array}{|c|c|c|}\hline 1 & 2^1 & \\ \hline & \mathbf{3^2} & \\ \hline \mathbf{4^2} & \mathbf{5^2} & \mathbf{6^2} \\ \hline \end{array}$$

where x^1 means that inserting or appending the cell labeled x is the first action performed during the current step, and the cells with a bold entry are the ones shifted down, this is the second action of the current step.

Note that the shape of the quasi-ribbon $P(\sigma)$ is the recoils composition of σ (that is the descents composition of σ^{-1}), and that $P(\sigma)$ is canonically labeled from left to right and from top to bottom. The labels of the cells of the ribbon tableau $Q(\sigma)$ record the positions in σ of the labels of the cells of $P(\sigma)$.

$$\boxed{1} \rightarrow \begin{array}{|c|}\hline 2 \\ \hline 1 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|}\hline 2 & \\ \hline 1 & 3 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline 2 & 4 & \\ \hline & 1 & 3 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|}\hline 2 & 4 & & \\ \hline & 1 & 3 & 5 \\ \hline \end{array} \rightarrow Q(\sigma) = \begin{array}{|c|c|c|}\hline 2 & 6 & \\ \hline & 4 & \\ \hline 1 & 3 & 5 \\ \hline \end{array}$$

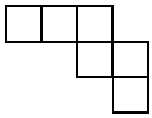
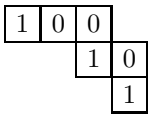
Now let us introduce the two dual graded graphs we will use to make the connection between the hypoplactic insertion algorithm and Fomin's approach for RSK.

2.1. **Dual graded graphs on compositions of integers.**

In this section, we are interested in isomorphic images of two graphs studied by Fomin, namely the *lifted binary tree* and *Binword* ([12], Example 2.4.1 and Fig.12). Their vertices are words on the alphabet $\{0,1\}$.

(1) in the *lifted binary tree*, a word w is covered by the two words $w.0$ and $w.1$ (where . denotes the usual concatenation of words), except 0 is only covered by 1 ;
(2) in *Binword*, there is an edge (u,v) if u is obtained by deleting a single letter (but not the first one) from v ; in addition there is an edge $(0,1)$.

Lemma 2.2. *There is a one-to-one correspondence between compositions of an integer n and n-letter words in the alphabet $\{0,1\}$, mapping a composition c to the word read after filling its diagram from left to right and top to bottom, with 1's in the first box and in any box following a descent position.*

$c = 321 \equiv$  ;  $\equiv w_c = 100101$

With this lemma, we can now build two dual graded graphs whose vertices of rank n are all the compositions of the integer n, with covering relations obtained by expressing the ones above on compositions of integers rather than on $\{0,1\}$-words. So in the first graph, a composition $c \neq \emptyset$ is covered by two compositions obtained either by increasing its last part, or by appending 1 after its last part ; in addition there is an edge $(\emptyset, 1)$. In the second graph, a composition $c \neq \emptyset$ is covered by the compositions obtained either by increasing a single part or by inserting a single 1, or by first splitting a single part into two parts and then increase one of the parts obtained, in addition there is an edge $(\emptyset, 1)$. This version of *Binword* appeared in [1].

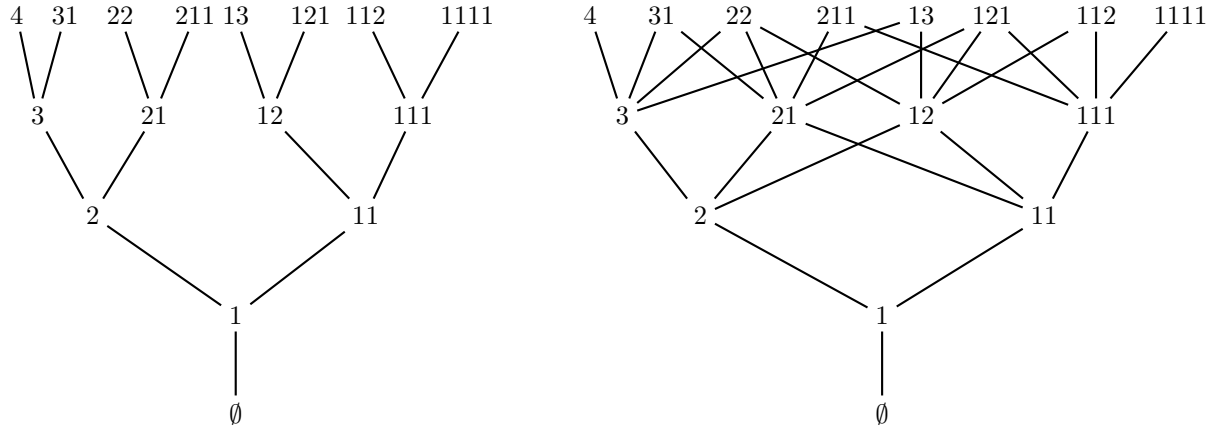


FIGURE 2. The lifted binary tree and *Binword* graphs on compositions of integers.

Proposition 2.1. *The following describes a* 1*-correspondence in the lifted binary tree and Binword as defined above on compositions on integers.*

Algorithm 1 A natural r-correspondence in the *lifted binary tree* and *Binword*.

if $t = x = y$ and $\alpha = 1$ **then**
  $z := x$, with its last part increased
else
  if $x = y$ **then**
    $z := x$, with an additional 1 at the end
  else
    if x ends with 1 **then**
      $z := x$, with an additional 1 at the end
    end if
  end if
end if

Proof. To show that this description is a 1-correspondence, one may consider the one described in ([14]-Lemma 4.6.1) and replace $y0$ by $y1$ wherever it appears, and conversely. One gets a second 1-correspondence ϕ in the lifted binary tree and Binword as defined on $\{0, 1\}$-words. Now apply Remark 2.2 to ϕ and you get the description above. □

In the next section, we show that using this 1-correspondence to build Fomin's growth diagram for any permutation σ, one gets two saturated chains that can be translated into a quasi-ribbon tableau $\widehat{P}(\sigma)$ and a ribbon tableau $\widehat{Q}(\sigma)$, and one naturally has $P(\sigma) = \widehat{P}(\sigma)$ and $Q(\sigma) = \widehat{Q}(\sigma)$.

2.2. **Growth diagram - equivalence of the two constructions.**

Let us build the growth diagram $d(\sigma)$ of the permutation $\sigma = 415362$, using the 1-correspondence defined in Proposition 2.1. On the upper boundary of $d(\sigma)$, one gets a satureted chain $\widehat{Q}$ in the *lifted binary tree*, and on the right boundary a chain $\widehat{P}$ in *Binword*. Fomin [15] gave a geometric construction for the Schensted insertion algorithm, Cameron and Killpatrick [9, 10] did the same for the Young-Fibonacci and the domino-Fibonacci insertion algorithms. We will now do the same for the hypoplactic insertion algorithm. It will consist in drawing shadow lines that can be used to directly determine the tableaux $P(\sigma)$ and $Q(\sigma)$ obtained through the hypoplactic insertion algorithm. To begin with, draw the permutation matrix of σ. Then from bottom to top, draw a set of broken lines to join the symbol $\mathbf{x}$ on the current line of the permutation matrix to the one on the line just above if the second symbol $\mathbf{x}$ is situated to the right of the first one, if this is not the case then start a new broken line with the second symbol $\mathbf{x}$. $P(\sigma)$ corresponds to reading vertical coordinates of the symbols $\mathbf{x}$ on the broken lines, from left to right and from bottom to top. As for $Q(\sigma)$, it corresponds to reading horizontal coordinates of the symbols $\mathbf{x}$ on the broken lines, in the same order.

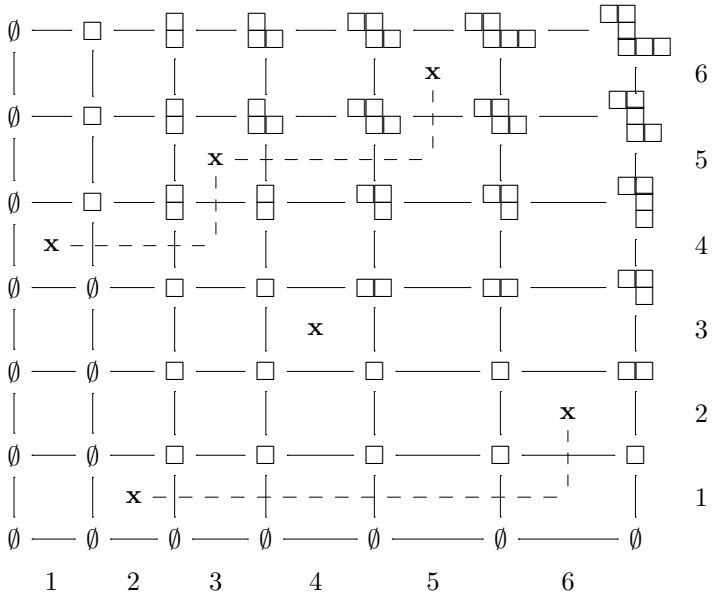


FIGURE 3. Example of growth diagram for the hypoplactic insertion algorithm.

Now one question which naturally presents itself is how to convert the chain $\widehat{P}$ into a quasi-ribbon tableau $\widehat{P}(\sigma)$ and the chain $\widehat{Q}$ into a ribbon tableau $\widehat{Q}(\sigma)$.

2.2.1. *From a chain of compositions to a standard quasi-ribbon tableau.*

Recall that in the chain $\widehat{P}$, a composition is covered by another one obtained either by increasing its last part, or by appending 1 after this last part, in addition $\emptyset$ is covered by 1.

$$\widehat{P} \;=\; \emptyset \;\to\; \mathbf{1} \;\to\; 2\mathbf{.} \;\to\; 21\mathbf{.} \;\to\; 21\mathbf{1} \;\to\; 21\mathbf{2} \;\to\; 213$$

So in order to get a quasi-ribbon tableau, one is simply to label the cells appearing in the right boundary of $d(\sigma)$ in the order they occur, and it is clear that this will always produce a quasi-ribbon tableau.

$$\emptyset \;\to\; \boxed{1^1} \;\to\; \boxed{1}\boxed{2^1} \;\to\; \begin{array}{|c|c|}\hline 1 & 2 \\ \hline \multicolumn{1}{c}{3^1} \end{array} \;\to\; \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 3 \\ \hline 4^1 \\ \hline \end{array} \;\to\; \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 \\ \hline 4 & 5^1 \\ \hline \end{array} \;\to\; \widehat{P}(\sigma) \;=\; \begin{array}{|c|c|c|}\hline 1 & 2 & \multicolumn{1}{c}{} \\ \hline 3 & \multicolumn{2}{c}{} \\ \hline 4 & 5 & 6^1 \\ \hline \end{array}$$

2.2.2. *From a chain of compositions to a standard ribbon tableau.*

Recall that in the chain $\widehat{Q}$, a composition is covered by another one obtained either by increasing a single part, or by appending 1 after any part (eventually just in front), in addition there is an edge $(\emptyset, 1)$.

$$Q \;=\; \emptyset \;\to\; 1\mathbf{.} \;\to\; 1\mathbf{1} \;\to\; \mathbf{1}2 \;\to\; \mathbf{2}\mathbf{2} \;\to\; 2\mathbf{.}3 \;\to\; 213$$

In order to get a ribbon tableau, we will process $\widehat{Q}$ as follows.

(1) when one part has been increased at step k, we append a cell labeled k just at the end of the corresponding line, and the lower part of the ribbon tableau is shifted to the right ;

(2) when 1 has been inserted after the i^{th} part, we shift down the portion of the ribbon tableau starting at the last cell on the i^{th} line, and then we insert a cell labeled k at this position.

It is clear that this will always produce a ribbon tableau.

$$\emptyset \;\to\; \boxed{1^1} \;\to\; \begin{array}{|c|}\hline 2^2 \\ \hline \mathbf{1}^1 \\ \hline \end{array} \;\to\; \begin{array}{|c|c|}\hline 2 & \multicolumn{1}{c}{} \\ \hline 1 & 3^1 \\ \hline \end{array} \;\to\; \begin{array}{|c|c|c|}\hline 2 & 4^2 & \multicolumn{1}{c}{} \\ \hline \multicolumn{2}{|c|}{\mathbf{1}^1} & \mathbf{3}^1 \\ \hline \end{array} \;\to\; \begin{array}{|c|c|c|c|}\hline 2 & 4 & \multicolumn{2}{c}{} \\ \hline \multicolumn{2}{|c|}{1} & 3 & 5^1 \\ \hline \end{array} \;\to\; Q(\sigma) \;=\; \begin{array}{|c|c|c|}\hline 2 & 6^2 & \multicolumn{1}{c}{} \\ \hline \multicolumn{2}{|c|}{\mathbf{4}^1} & \multicolumn{1}{c}{} \\ \hline \mathbf{1}^1 & \mathbf{3}^1 & \mathbf{5}^1 \\ \hline \end{array}$$

Proposition 2.2. *For any permutation σ, $P(\sigma) = \widehat{P}(\sigma)$ and $Q(\sigma) = \widehat{Q}(\sigma)$.*

Proof. This follows from the definition of the hypoplactic insertion algorithm, and the remark that any composition $\widehat{Q}_k$ appearing in $\widehat{Q}$ is the shape of the tableau $P(\sigma_1\sigma_2\cdots\sigma_k)$ where $\sigma_1\sigma_2\cdots\sigma_k$ is the restriction of σ to its first k letters, and any composition $\widehat{P}_k$ appearing in $\widehat{P}$ is the shape of $P(\sigma_{/[1..k]})$ where $\sigma_{/[1..k]}$ is the restriction of σ to the interval $[1..k]$. Indeed, with this remark, the canonical labelling of $P(\sigma)$ coincides with the description of the conversion of $\widehat{P}$ into $\widehat{P}(\sigma)$. As for $\widehat{Q}$, its conversion into $Q(\sigma)$ clearly coincides with the description of the hypoplactic insertion algorithm (see Example 2.1). □

3. Fomin's approach applied to the sylvester insertion algorithm

3.1. **The insertion algorithm.**

The sylvester insertion algorithm is a variant of the BST [4]. Introduced by Hivert et al [6], it was used to give a new construction in term of noncommutative polynomials, of the algebra of Planar Binary Trees of Loday-Ronco [8]. Trough this algorithm, the insertion tree of a given permutation σ is the binary search tree built by *reading σ from right to left*. The recording tree is a *decreasing tree*, that is to say a labeled binary tree such that the label of each internal node is greater than the labels of all the nodes in its subtrees. It records the positions in σ of the labels of the insertion tree.

In the classical version of the BST, permutations are *read from left to right* and the insertion tree will be denoted $\mathbb{P}(\sigma)$. The recording tree $\mathbb{Q}(\sigma)$ is an *increasing tree*, that is to say a labeled binary tree such that the label of each internal node is smaller than the labels of all the nodes in its subtrees. So for $\sigma = 351426$, one gets the following insertion and recording trees.

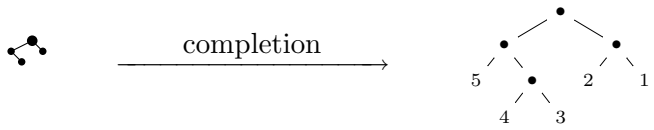
$$\mathbb{P}(\sigma) \;=\; \begin{array}{ccccccc} & & & 3 & & & \\ & & 1 & & & 5 & \\ & & & 2 & 4 & & 6 \end{array} \qquad ; \qquad \mathbb{Q}(\sigma) \;=\; \begin{array}{ccccccc} & & & 1 & & & \\ & & 3 & & & 2 & \\ & & & 5 & 4 & & 6 \end{array}$$

In section 3.3 we show how Fomin's approach applies to the BST, but let us first introduce the corresponding dual graded graphs.

3.2. **Dual graded graphs on binary trees.**

Once again, we will consider isomorphic images of two dual graded graphs studied by Fomin, namely the *lattice of binary trees* and the *bracket tree* ([12], Fig.13). The *lattice of binary trees* is defined as follows. Its vertices of rank n are the syntactically correct formulae defining different versions of calculation of a non-associative product of $n+1$ entries. So any vertex of rank n is a valid sequence of $n-1$ opening and $n-1$ closing brackets inserted into $x_1\,x_2\cdots x_n$. In the *bracket tree*, two expressions are linked if one results from the other by deleting the first entry, and then removing subsequent unnecessary brackets, and renumbering the new expression. In the sequel we will be considering the *reflected bracket tree*, the graph where two expressions are linked if one results from the other by deleting the last entry, and then removing subsequent unnecessary brackets, and renumbering the new expression.

Lemma 3.1. *There is a one-to-one correspondence between unlabeled binary trees and bracketed expressions. A tree t is identified with the expression obtained by completion of t, adding one leaf to any node having a single child-node, and two leaves to any childless node. Then label the n leaves of the resulting binary tree t' according to the right-to-left infix order, using each of the labels $1, 2, \cdots, n$ only once. If t' is empty then the bracketed expression is x_1, else the bracketed expression is obtained by recursively reading its right and left subtrees. Below is an example where the expression obtained is $(x_1x_2)((x_3x_4)x_5)$.*



With this lemma, we can build two graphs whose vertices of rank n are all the binary trees having n nodes, with covering relations obtained by expressing the ones above on binary trees rather than on bracketed expressions. So one will have the following covering relations.

(1) in the *lattice of binary trees*, any tree is covered by all those obtained from it by addition of a single node, in all possible ways. Below is a finite realization of the graph, from rank 0 to rank 4.

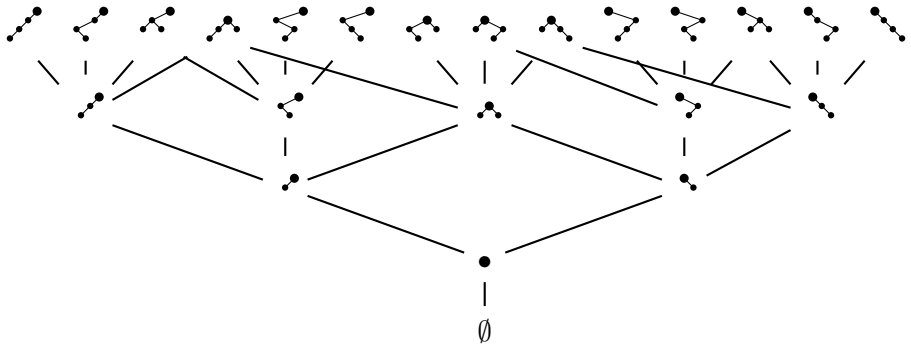


FIGURE 4. The lattice of binary trees.

(2) in the *reflected bracket tree*, a tree y covers a single tree t obtained from y by deleting it's right-most node if any, or its root otherwise, and replacing the deleted node by its own left subtree if any. Below is a finite realization of the graph, from rank 0 to rank 4.

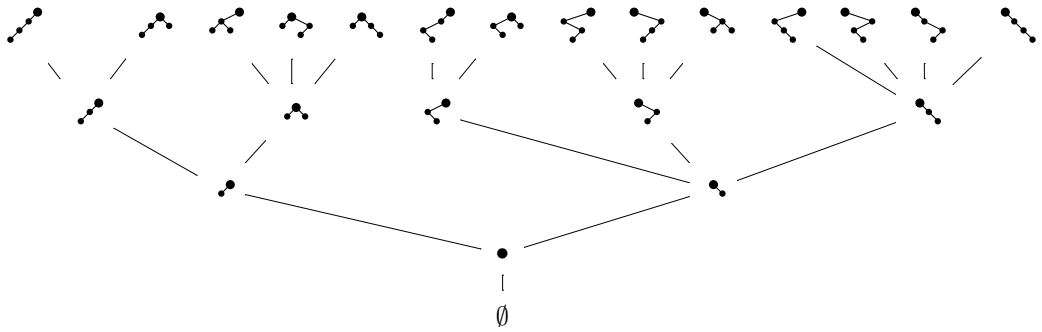


FIGURE 5. The reflected bracket tree, a dual of the lattice of binary trees.

Remark 3.1. *The reader should pay attention that for a more convenient graphical representation, vertices at rank* 4 *are not listed in the same order in the two graphs above.*

Proposition 3.1. *The reflected bracket tree is dual to the lattice of binary trees.*

Proof. Follows from the duality of the lattice of binary trees and the bracket tree. □

Proposition 3.2. *The following algorithm describes a* 1*-correspondence in the* lattice of binary trees *and the* reflected bracket tree*.*

Algorithm 2 natural r-correspondence in the *lattice of binary trees* and the *reflected bracket tree.*

if $t = x = y$ and $\alpha = 1$ **then**
  $z := t$, with one node added as right child-node of its rightmost node
else
  if $x = y$ **then**
    $z := y$, with one node added as left child-node of its rightmost node
  else
    $z := x$, with one node added in such a way that deleting the right-most one gives back y [1]
  end if
end if

[1] Due to the duality of the two graphs, there is one and only one way doing this.

Remark 3.2. *This algorithm does not apply to degenerated cases for which* z *is trivially deduced from the definition of a* 1*-correspondence.*

3.3. **Growth diagram - equivalence of the two constructions.**

Let us build the growth diagram $d(\sigma)$ of the permutation $\sigma = 351426$. On the upper boundary of $d(\sigma)$, one gets a saturated chain $\widehat{\mathbb{Q}}$ in the *lattice of binary trees*, and on the right boundary a chain $\widehat{\mathbb{P}}$ in the *reflected bracket tree*.

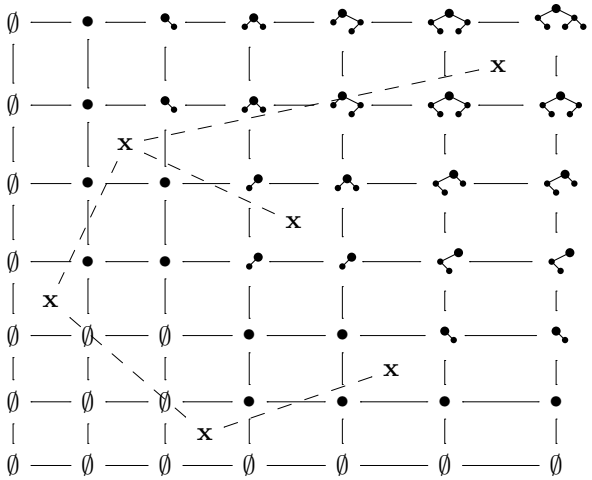
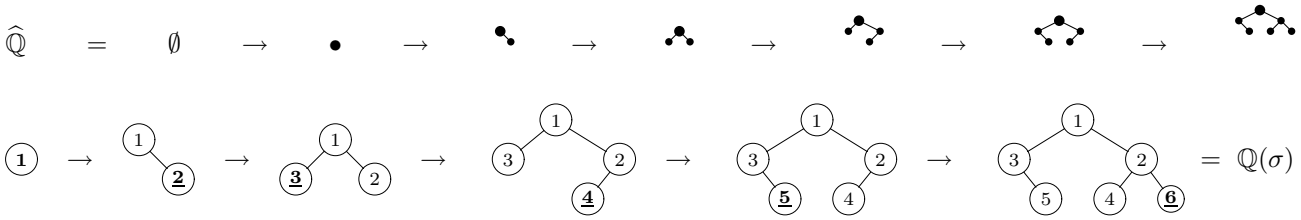


FIGURE 6. Example of growth diagram for the BST insertion algorithm.

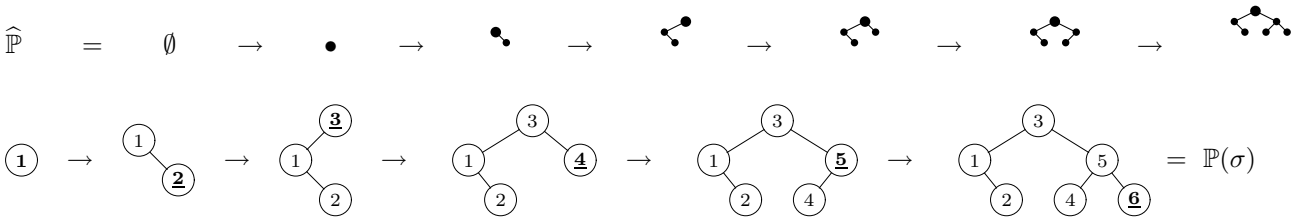
3.3.1. *From a chain of binary trees to an increasing tree.*

Converting the horizontal chain $\widehat{\mathbb{Q}}$ into an increasing tree is naturally done by labeling its trees according to the order in which their appear in the path.



3.3.2. *From a chain of binary trees to a binary search tree.*

In the vertical chain $\widehat{\mathbb{P}}$, each tree $\widehat{\mathbb{P}}_i \neq \emptyset$ differs from its predecessor $\widehat{\mathbb{P}}_{i-1}$ by the node $\bullet_i$ that should be deleted and replaced by its own left subtree (in any) in order to obtain $\widehat{\mathbb{P}}_{i-1}$. Label this node $\bullet_i$ with i, then the remaining nodes form a tree of shape $\widehat{\mathbb{P}}_{i-1}$ whose nodes labels where set during the previous step of the conversion.



Proposition 3.3. *For any permutation σ, $\mathbb{P}(\sigma) = \widehat{\mathbb{P}}(\sigma)$ and $\mathbb{Q}(\sigma) = \widehat{\mathbb{Q}}(\sigma)$.*

Proof. That building the growth diagram of a permutation σ is a parallel version of the application of the binary search tree insertion algorithm to σ follows from the remark that any tree $\widehat{\mathbb{Q}}_k$ appearing in $\widehat{\mathbb{Q}}$ is the shape of the binary search tree $\mathbb{P}(\sigma_1\sigma_2\cdots\sigma_k)$ where $\sigma_1\sigma_2\cdots\sigma_k$ is the restriction of σ to its first k letters, and any tree $\widehat{\mathbb{P}}_k$ appearing in $\widehat{\mathbb{P}}$ is the shape of $\mathbb{P}(\sigma_{/[1..k]})$ where $\sigma_{/[1..k]}$ is the restriction of σ to the interval $[1..k]$. □

References

[1] *A. Björner* and R. P. Stanley, An analogue of Young's lattice for compositions, *arXivmath.CO/0508043*.

[2] *C. Schensted*, Longest increasing and decreasing subsequences. *Canad. J. Math.*, vol. 13, 1961, pp. 179-191.

[3] *D. E. Knuth*, Permutations, matrices and generalized Young tableaux, *Pacific J. Math.* 34 (1970) 709–727.

[4] *D. E. Knuth*, The art of computer programming, vol.3: *Searching and sorting* (Addison-Wesley, 1973).

[5] *D. Krob* and *J.-Y. Thibon*, Noncommutative symmetric function IV: Quantum linear groups and Hecke algebras at q=0, *J. Alg. Comb.* 6 (1997), 339-376.

[6] *F. Hivert*, *J. C. Novelli*, and *J.-Y. Thibon*, The Algebra of Binary Search Trees, *Theo. Comp. Science* 339 (2005), 129-165.

[7] *J. Nzeutchap*, Dual Graded Graphs and Fomin's r-correspondences associated to the Hopf Algebras of Planar Binary Trees, Quasi-symmetric Functions and Noncommutative Symmetric Functions, *Best Poster Award - FPSAC'06.*

[8] *J.-L. Loday* and *M. O. Ronco*, Hopf Algebra of the Planar Binary Trees, *Adv. Math*, 139 (1998) n. 2, 293-309.

[9] *K. Killpatrick*, Evacuation and a Geometric Consturction for Fibonacci Tableaux, *J. Comb. Th, Series A* 110 (2005), 337-351.
[10] *N. Cameron* and *K. Killpatrick*, Domino Fibonacci tableaux, *Elec. Jour. of Comb.* 13 (2006), #R45.
[11] *R. P. Stanley*, Differential Posets, *J. Amer. Math. Soc.* 1 (1988), 919-961.
[12] *S. Fomin*, Duality of Graded Graphs, *J. Alg. Comb.* 3 (1994), 357-404.
[13] *S. Fomin*, Generalized Robinson-Schensted-Knuth correspondence, *Zapiski Nauchn. Sem. LOMI.* 155 (1986), 156-175.
[14] *S. Fomin*, Schensted Algorithms for Dual Graded Graphs, *J. Alg. Comb.* 4 (1995), 5-45.
[15] *T. Britz* and S. Fomin, Finite posets and Ferrers shapes, *Adv. Math.* 158 (2001), 86-127.
[16] *T. Roby*, Applications and extensions of Fomin's generalization of the Robinson-Schensted correspondence to differential posets, *Ph.D. thesis, MIT,* 1991.
[17] *T. Roby*, The connection between the Robinson-Shensted correspondence for skew oscillating tableaux and graded graphs, *Descrete mathematics* 139 (1995), 481-485.

LITIS EA 4051 (Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes), Avenue de l'Université, 76800 Saint Etienne du Rouvray, France
E-mail address: janvier.nzeutchap@univ-mlv.fr
URL: http://monge.univ-mlv.fr/~nzeutcha